# 八皇后问题

## 经典八皇后问题

在国际象棋种，皇后是最强大的一枚棋子，可以吃掉与其在同一行，同一列和斜线上的敌方棋子。

八皇后问题就是将八个皇后摆在一张8\*8的国际象棋棋盘上，使每个皇后都无法吃掉别的皇后，一共有多少种摆法？

这个问题是典型的回溯法解决问题。

来自刘汝佳《算法竞赛入门经典》的算法：

void search(int row){

if(row==n) count++;//递归边界。只要走到了这里，所有皇后必然不冲突。

else for(int i=0;i<n;i++){

C[row]=i;//尝试把第row行的皇后放在第i列。

int ok=0;

for(int j=0;j<row;j++)//检查是否和前面的皇后冲突

if(C[row]==C[j]||row-C[row]==j-C[j];row+C[row]=j+C[j])//判断条件

{ok=0;break; }

If(ok==2) search(row+1);

}

}

因为

因此只要传入皇后的个数，即可算出有多少种摆法。

那么判断条件为什么是这样呢？

因为是按行摆放，所以同行就不用考虑了。

而不能同列的判断条件就是和前面已经存在的行依次做比较：C[row]==C[j]

而不能同斜线的判断条件就是和已经存在的行依次做如下比较：

row-C[row]==j-C[j];row+C[row]=i+C[i]

行和列都比较好理解。关键是斜线的判断条件问什么是这样呢。

其实斜线可以分为主对角线和副对角线。

如果我们把整个棋盘看作是一个作标系。每个点的坐标为（row,col），每一条主对角线的斜率都是1，副对角线的斜率都是-1。所以每一条主对角线的函数公式为，col=row+t。所以每一条主对角线上的格子都满足，col-row=t，从而得出若row-C[row]==j-C[j]这个条件为真，则格子（row,C[row]）必然在一条对角线上;而副对角线上的函数方程为col=-row+t，同上可以推出row+C[row]==j+c[j]的判断条件。

只要新进入的行满足上诉条件，就说明一定不会有问题。否则就回溯到上一个条件，选择下一个可能存放的位置。

以后需到这种在固定区域的判断问题，可以尝试利用建立坐标系，然后建立函数公式，再来计算条件的方式来解决问题。

## 经典八皇后问题的扩展（1）

若已知八皇后问题的解是92。

输入：

第一行是一个整数n,代表接下来要输入的整数个数，1<=n<=92。

接下来n行每行一个整数k，代表了把八皇后问题的第k个解。（解的形式是一个8位整数，从左到右分别为第一行到第八行的列序号。根据这些列序号组成的8位整数的大小来排序）。

输出：

输出n行，每行一个整数，和输入的n个数一一对应，以8位整数的形式输出输入的数代表的解。

解题思路：求出8皇后问题的所有8位数解，存放在一个list中，然后调用Collections的sort方法排序这个list。再遍历输入数组，依次输出对应的解。解题代码如下：

**package** shunfeng;

**import** java.util.ArrayList;

//

**import** java.util.Collections;

**import** java.util.Scanner;

//八皇后的问题

**public** **class** EngihtQueens {

**static** **int** *b*[]=**new** **int**[9];

**static** **int** *count*=1;

**public** **static** **void** main(String[] args) {

Scanner in = **new** Scanner(System.***in***);

**int** n=in.nextInt();

ArrayList<Integer> list=**new** ArrayList<Integer>();

**int** a[]=**new** **int**[n];

**for**(**int** i=0;i<n;i++){

**int** temp=in.nextInt();

a[i]=temp;

}

*queen*(0,list,0);

Collections.*sort*(list);

**for**(**int** i=0;i<n;i++) {

System.***out***.println(list.get(a[i]-1));

}

}

**public** **static** **void** queen(**int** n,ArrayList<Integer> list, **int** index){

**if**(index==8){

list.add(n);

*count*++;

**return** ;

}

**for**(**int** i=0;i<8;i++){

*b*[index]=i;

**if**(*is\_OK*(index))

*queen*(n\*10+i+1,list,index+1);

}

}

**public** **static** **boolean** is\_OK(**int** index) {

**for**(**int** i=0;i<index;i++) {

**if**(*b*[index]==*b*[i]||index-*b*[index]==i-*b*[i]||index+*b*[index]==i+*b*[i])

**return** **false**;

}

**return** **true**;

}

}